Zusammenfassung TI I

Boole'sche Algebra:

$A \cdot B = B \cdot A$	$A + B = B + A$	$A \cdot A = A$	$A + A = A$
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$	$A \cdot 0 = 0$	$A + 1 = 1$
$A \cdot (B + C) = A \cdot B + B \cdot C$	$A + B \cdot C = (A + B)(A + C)$	$A \cdot (A + B) = A$	$A + (A \cdot B) = A$
$A \cdot 1 = A$	$A + 0 = A$	$\bar{\bar{A}} = A$	
$A \cdot \bar{A} = 0$	$A + \bar{A} = 1$	$\overline{A \cdot B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \cdot \bar{B}$

DNF/KNF: allg. Form:

DNF: $y = a_0 \bar{x}_3 \bar{x}_2 \bar{x}_1 + a_1 \bar{x}_3 \bar{x}_2 x_1 + \dots + a_7 x_3 x_2 x_1$

KNF: $y = (a_0 + x_3 + x_2 + x_1) \cdot (a_1 + x_3 + x_2 + \bar{x}_1) \cdot \dots \cdot (a_7 + \bar{x}_3 + \bar{x}_2 + \bar{x}_1)$

Minimierungsverfahren (McCluskey):

(i) Aufstellen der DNF

(ii) Bestimmung d. Primterme: Aufteilung in Klassen K_i ($i \hat{=}$ Anzahl d. nichtnegierten Variablen) + $F = 1$

↳ Zusammenfassung v. Termen aus benachbarten Klassen

(iii) Bestimmung d. wesentlichen Primterme: → Primterm, der einen Minterm exklusiv abdeckt

(iv) Bestimmung d. Restmatrix, bestehend aus allen nicht wesentlichen Primtermen und Mintermen, die nicht von Primtermen abgedeckt werden

Schaltalgebraische Gleichungen: allg. Form:

$f_1(A_1, \dots, A_n; x_1, \dots, x_k) = g_1(A_1 \dots A_n; x_1 \dots x_k)$

$\vdots$

$f_m(\dots) = g_m(\dots)$

→ Systematischen Prüfen aller Möglichkeiten

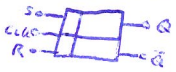
Lösungsausdruck: (1) es ex. genau ein Lsg-Vektor + dessen Minterm

(2) mehrere Lsg. → Freiheitsgrade

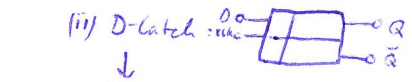
(3) keine Lsg → Verträglichkeitsbedingung: $\sum_{\mu} m_\mu = 0$

Flip-Flops: Arten:

- Latch (Taktzustandsgesteuert): (i) SR-Latch

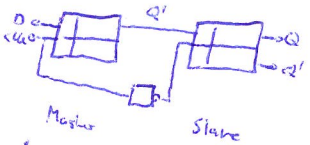


(ii) D-Latch



(CLK = 1 → Q = D
CLK = 0 → Q = D |fallende Flanke)

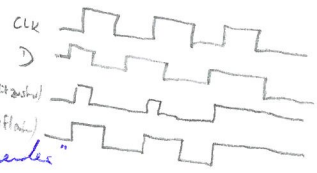
- Master-Slave-FF:



↳ Aufbau allg. synchr. Schaltwerke möglich

- Taktflankengesteuertes FF:

Wie M.-S.-FF, nur bei „steigender" Taktflanke



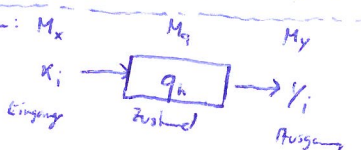
~~Eingangs-FF (XXXX):~~

Charakt. Schaltfkt.:

- D-FF: $Q^{n+1} = D^n$
- SR-FF: $Q^{n+1} = (S + \bar{R}Q)^n$
- T-FF: $Q^{n+1} = (\bar{T}Q + T\bar{Q})^n$
- JK-FF: $Q^{n+1} = (J\bar{Q} + \bar{K}Q)^n$ ... wie SR-FF mit Toggle-Fkt
- Enable-D-FF: $Q^{n+1} = (ED + \bar{E}Q)^n$

Automatentheorie:

Mengen: M_x, M_q, M_y

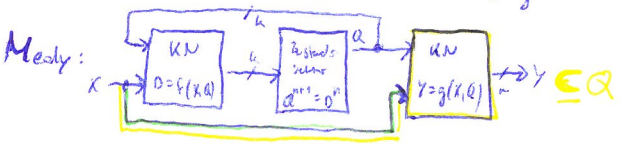


Übergangsfkt: $f: M_x \times M_q \to M_q$

Ausgangsfkt: $g: M_x \times M_q \to M_y$ (Mealy)

$g: M_q \to M_y$ (Moore)

Mealy:



(fällt weg bei Moore

Entwurf sequentieller Netzwerke

- Kombinatorisches Netzwerk ≙ Rückkopplungsfreie digitale Schaltung ohne Speicherelemente
- Sequentielles Netzwerk ≙ Digitale Schaltung mit Speicherelementen
- Schaltungstechnische Realisierung:
  - Eingangs-/Ausgangssignale, Zustände: binär codierte Boolesche Vektoren
  - Übergangsfkt: $Q^{n+1} = f(X^n, Q^n)$ $\xrightarrow[\text{Speicher-netzwerk (k D-FF)}]{Z^{n+1}=D}$ $D^n = f(X^n, Q^n)$ (kombinat. Netzwerk)
  - Ausgangsfkt.: rein kombinatorisches Netzwerk
- Systemat. Vorgehen:
  (i) Zustandsübergangsdiagramm
  (ii) Zustandsreduktion (selten möglich)
  (iii) Codierung d. Zeichen/Zustände
  (iv) Erweiterte Zustandsübergangstabelle
  (v) Aufstellen d. Problemfkt.
  (vi) Wahl d. Speicherelemente
  (vii) Bestimmung d. Eingangsfkt.
  (viii) Bestimmung d. Ausgangsfkt.
  (ix) Schaltplan!

Zahlensysteme & -darstellungen

$1100 \;\; 1001 = 153_{10} = 201_{16}$ (C, 9)

- Stellenwertsystem: $N_z = \sum_{i=0}^{n-1} a_i \cdot z^i$ mit: $a_i \in \{0,1,\dots,z-1\}$; $z \hat{=}$ ganzzahlige Basis; $n \hat{=}$ Stellenzahl; $N_z \hat{=}$ Wert
- 2-Komplement: $K_z = z^n - |N_z| = \sum_{i=0}^{n-1} \left(\underbrace{(z-1-a_i)}_{\text{größte darstellbare Ziffer}} z^i \right) + 1$ (VZ-behaftete und VZ-lose Zahlen in "Zweierkomplement-Arithmetik" einheitlich betrachtet werden)
- Wertebereich: VZ-los: $0 \le x \le z^n - 1$
  Überschreitung d. Wertebereichs: Übertrag $c = c_n$ aus höchstwertiger Stelle $n-1$
  VZ-behaftet: Überlauf $V = c_{n-1} \oplus c_n$ (auch bei Subtraktion)
  geschlossene Darstellung: $a = -a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$
  VZ-richtige Erweiterung: VZ-Bit kopieren

  Stellenanzahl: $[a_1, a_2]$, $n = \lceil \ln(a_2 - a_1) \rceil$
- Gebrochene Zahlen:
  Festkommazahlen: analog wie oben: $x = \sum_{i=-m}^{n} a_i z^i$
  Fließkommadarstellung: $x = a \cdot 2^e = (1+f) 2^e$ mit a ... normierte Mantisse, $1 \le a < 2$; f ... Fraktion, $0 \le f < 1$; e ... Exponent, ganzzahlig

  ≙ x_{GD}: | S | E | F |

  Codierung: f codiert als Festkommazahl mit 0 vor- und m Nachkommastellen
  e wird „Bias" B (Offset) addiert, s.d. $e+B > 0$
  $E_2 = E_{10} + B_{10}$ ← Codierung d. Exponenten mit Bias als VZ-lose Binärzahl mit k Stellen (i.d.R.: $B = 2^{k-1} - 1$)

  Single Precision: $B = 127$, $p = 23$, $k = 8$
  double precision: $B = 1023$, $p = 53$, $k = 11$

ALU

Addierer

Halbaddierer: HA (x, y → c, s)

$s = x \oplus y$

$c = x \cdot y$

$t = 2\tau$

Volladdierer: VA (c_{in}, x, y → c_{out}, s)

(analog: VS)

$s = \bar{x}\bar{y}c_{in} + \bar{x}y\bar{c}_{in} + x\bar{y}\bar{c}_{in} + xyc_{in}$

$c_{out} = y \cdot c_{in} + x \cdot c_{in} + xy$

$t = 2\tau$

Ripple-Carry: VA-Kette (c_{out} ← c_2 ← c_1 ← c_0; s_{n-1}, s_1, s_0)

$\tau_{min} = 2\tau$ $\tau_{max} = n \cdot 2\tau$

Carry-Look-Ahead: 2 Fälle: (i) $x_i \equiv y_i \Rightarrow c_{i+1}$ unabh. von c_i

(ii) $x_i \not\equiv y_i \Rightarrow c_{i+1} = \begin{cases} 1 & \text{wenn } x_i y_i = 1 \\ c_i & \text{wenn } x_i \oplus y_i = 1 \\ 0 & \text{sonst} \end{cases}$

$\Rightarrow c_{i+1} = G_i + P_i c_i$ mit $G_i = x_i y_i$

$P_i = x_i \oplus y_i$

Zusammenfassung TI I

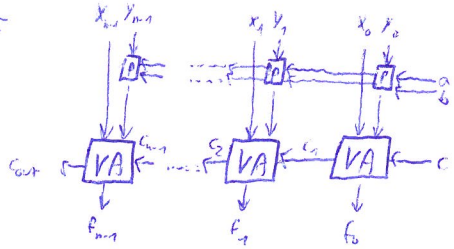
ALU

Einstellige Logikfunktionen

a	b	$\psi(x) = ax + b\bar{x}$
0	0	0 (Nullfkt.)
0	1	$\bar{x}$ (Inversion)
1	0	x (Identität)
1	1	1 (Einsfkt.)

Arithmet. Einheit (AU)

a	b	c	$\psi(y_i)$	arithmet. Interpretation
0	0	0	0	x
0	0	1	0	$x+1$
0	1	0	$\bar{y}_i$	$x-y-1$
0	1	1	$\bar{y}_i$	$x-y$
1	0	0	y_i	$x+y$
1	0	1	y_i	$x+y+1$
1	1	0	1	$x-1$
1	1	1	1	x



Arithmet.-log. Einheit (ALU)

d	b	a	Fkt.
0	0	0	x OR y
0	0	1	x XOR y
0	1	0	x AND y
0	1	1	NOT x
1	0	0	x+c
1	0	1	x+y+c
1	1	0	x-y-$\bar{c}$
1	1	1	x-$\bar{c}$

(Ansatz: ~~Mehrfach-Logik~~ weitere Steuerschaltung parametr. Vorbereitungsschaltung vor Eingang $x \rightarrow \psi(x)$)

Booth: z.B. 01101101 (muss hinzugefügt werden) $d_k = -2a_{2k+1} + \bar{a}_{2k} + a_{2k-1}$

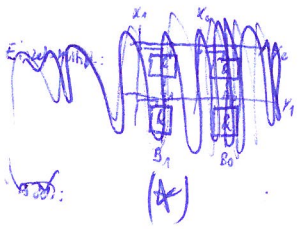
$d_3\ d_2\ d_1$

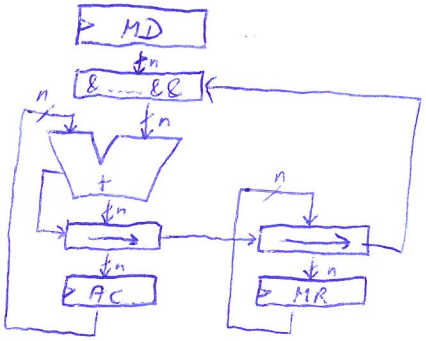
d_{dez}: $01101_2 = 12 - 11_{Booth}$ mit $d_k = \begin{cases} 0 \dots \text{nix} \\ \pm 1 \dots \text{stellenrichtige} \\ \pm 2 \dots \text{1 Stelle links} \end{cases}$ stellen. sub.

beachte: um 2 schieben!

Multiplikation & Division

Parallel-Multiplikationswerke





MD ... Multiplikand-Register (unverändert)
MR ... Multiplikator-Register (niederwertige n Bit d. Erg.)
AC ... Akkumulator-Register (höherwertige n Bit d. Erg.)

(auch rein kombinatorisch möglich)

VZ-behaftete Multiplikation:
- VZ-richtige Erweiterung
- höchstwertiges Teilprodukt: negieren!

Dividierwerke: → Restoring-Algorithmus (175 ff)
→ Nonrestoring-"

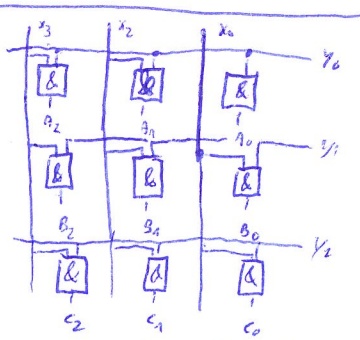
Elemente & Mechanismen

d. Register-Transfer-Ebene

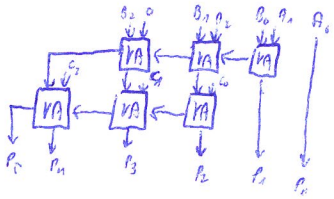
Hardware-Entwurfsebenen:

Ebene	Schaltkreis	Gatter-/Logik	RT	Microprogramm	Assembler
Elemente	Dioden, Transistoren, ...	Gatter, FF	Operatore, Multiplexer, ALU's, Steuerwerke, ...	Register, Datenpfade, Operatoren	Register, Speicherzellen
Zeitverhalten	ns, ps	Gatterlaufzeiten	Taktperiode	Taktperiode	mehrere Taktperioden (Sprung)
Informationseinheiten	Spannung/Strom	Bits	Worte	Worte	Worte
Strukturbeschreibung	Schaltplan, Chiplayout	Schaltplan, Netzliste	Blockschaltbild, VHDL	Blockschaltbild, VHDL	Programmiermodell, Speichermodell
Verhaltensbeschreibung	mathematisch, z.B. Dgl.	schaltalgebraische Gleichungen	Evtl. Automaten, Ablaufdiagramme, Microprogramme, VHDL	Microcode-Assembler-sprache	Assembler-sprache
Methoden	Maschen-/Knotenanalyse, Graphentheorie, ...	Axiome/Sätze d. Schaltalgebra, rechnergestützte Optimierungsverfahren	Entwurf digitaler Systeme, VHDL	Microcode-Programmierung	Assembler-programmierung

A) Einzelmultipl.:



Addition:



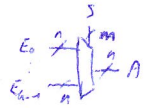
Elemente d. RT-Ebene:
- Zusammenfassung Boolsche Größen → Boolsche Vektoren
- Daten- & Steuersignale (bei Steuersignale: (i) Festverdrahtete Steuerwerke
(ii) Microprogrammierte Steuerwerke)

Datentore:

(Tristatebuffer)

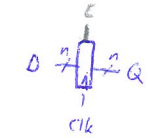
G = 0: Ausgänge werden physikal. von Eingängen getrennt
G = 1: Ausgänge werden mit Eingängen verbunden

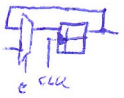
Datenselektoren: (Multiplexer)



dient dazu, in Abhängigkeit von einem/mehreren Auswahleingängen einen von mehreren Eingangsbusse auf den Ausgangsbusse durchzuschalten

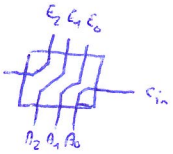
Register:
- Speicherelemente
- steuerbar / nicht-steuerbar (enable-flag)
- z.B.



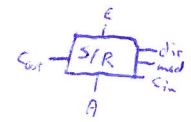


Kombinatorische Schiebe-/Rotierwerke:

(i) Festverdrahtete Schiebewerke:

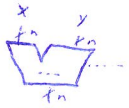


(ii) Einstellbare Schiebe-/Rotierwerke:
(ist Anzahl d. Stellen, um die geschoben wird, einstellbar: Barrel-Shifter)

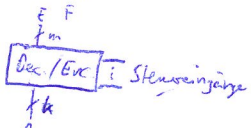


mod	dir	Fkt.
0	0	links schieben
0	1	rechts "
1	0	links rotieren
1	1	rechts "

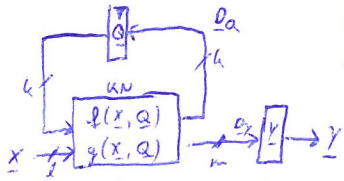
Arithmet. Elemente: Addierer, Subtrahierer, ...



Dekoder/Encoder:
- k ≥ m: Dekoder
- k < m: Enkoder



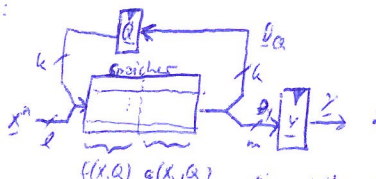
Mikroprogrammierung: Festverdrahtetes Steuerwerk:



Q: Register aus k D-FF
Y: Register aus m FF
KN: $Q^{n+1} = D_Q^n = f(X^n, Q^n)$
$Y^{n+1} = D_Y^n = g(X^n, Q^n)$

Eigenschaften: Funktionalität fest verdrahtet (→ nicht programmierbar)

Einfaches mikroprogr. Steuerwerk:



Q: s.o.
Y: s.o.
Speicher: 2^{lk} Adressbits, 2^{lk} Worte à k+m Bit

Eigenschaften:
- für gegebene X_i & Q_j sind $f(X_i, Q_j)$ und $g(X_i, Q_j)$ im Speicher an Adresse (X_i, Q_j) abgelegt
- Speicherinhalt und somit Fkt. kann geändert werden
- Speicherkapazität: $2^{(l+k)}(m+k)$

Möglichkeiten zur Steuersignalcodierung:

Microoperation: elementare 1-Takt-Operation
Microbefehl: Microoperationen, die parallel in 1 Takt ausgeführt werden
Microcodierung: Codierung d. Microbefehle
Microprogramm: Sequenz von Microbefehlen

Horizontaler Microcode:
- jedes Bit ≙ 1 Steuerleitung
- Microoperationen eines Microbefehls werden parallel ausgeführt / horizontal codiert
↳ sehr breite Codewörter

Vertikaler Microcode:

Zusammenfassung TI I

Vertikaler Mikrocode:
- Mikrobefehl realisiert nur eine Mikrooperation oder eine Kombination voneinander abh. Mikrooperationen
- Steuersignale werden durch Codierung d. Mikrocodeworts erzeugt (→ Decoder notwendig)
- eingeschränkte Parallelausführung: Mikrooperationen werden sequentiell ausgeführt
- kurze Mikrocodewörter

Minimalcodierter m: Gruppen sich gegenseitig ausschließender Mikrooperationen werden vertikal codiert
- Diese Gruppen und restl. Mikrooperationen werden horizontal codiert

↳ Elimination unbenötigter Freiheitsgrade
↳ Reduktion d. Codewortlänge

Möglichkeiten zur Ablaufsteuerung: Problem: * Bei Steuerung komplexer digitaler Systeme (z.B. Mikroprozessor) werden große Mikrocodespeicher benötigt

↳ **Multiplexen d. Eingangssignale:** * weniger mögliche Folgezustände
↳ drastische Reduzierung d. Speicherplatzbedarfs

Erweiterungen:
- zusätzl. Eingänge mit konstant 1 und 0 → unbedingte Folgeadresse
- Vermeidung von 1 Multiplexer
- Bedingungsdecoder (anstatt Multiplexer)

Automat. Ermittlung sequentieller Folgeadressen:
- Halbierung d. Speicherbedarfs (-1 Adressbit)
- Wahl zw. automat. inkrementierter Adresse und explizitem Sprungziel abhängig von ausgewähltem Eingangssignal

Verwendung eines Auswahlfelds:
- Auswahlfeld entscheidet, wie das restl. Speicherwort interpretiert wird (Folgeadresse oder Steuersignale)

Weitere Möglichkeiten:
- Zähler für Schleifen im Mikrocode
- Flut-Blöcke zur Realisierung von Mikrocode-Unterprogrammen (Stapel)
- Mgl. zum Aufruf von Mikroprogrammen von außen
- Segmentierung d. Mikroprogrammspeichers

Microprogrammierter Prozessor:

Speicher:
- enthält Programmcode (Besteht aus Maschinenbefehlen) und Daten

Befehlsregister (IR): wird mikroprogrammgesteuert jeweils mit dem nächsten auszuführenden Maschinenbefehl geladen

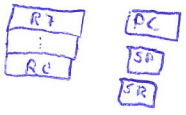
Befehlsdecoder: ermittelt aus dem Code d. Maschinenbefehls die Start-adresse d. zugehörigen Mikroprogramms im Speicher

Steuerwerk:
- führt Mikroprogramm an der vom Befehlsdecoder angegebenen Startadresse Schritt für Schritt aus
- Mikroprogramm ist verantwortlich dafür, dass nächster Maschinenbefehl am Ende geladen & dekodiert wird

Assemblerebene

Programmiermodell (ISA): Umfasst den Befehlssatz und alle Elemente, auf die Programme mit Hilfe von Befehlen zugreifen kann (Register innerhalb d. Prozessors (Registersatz) und Speicherworte in einem externen Speicher (Speichermodell))
(Mikroarchitektur beschreibt im Gegensatz zur ISA die tatsächliche Implementierung auf d. RT-Ebene)

Registersatz: Bsp.:



mit R_i ... Allzweckregister i = 0, ..., 7
PC ... Program Counter
SP ... Stack Pointer
SR ... Status Register

Befehlstypen

Typ	Art	Bsp.
Datenübertragungsbefehle	Speicher ⇔ Reg. Reg. ⇔ Reg. Speicher ⇔ Speicher	MOVE
Datenbearbeitungsbefehle	arithmet. Befehle log. Befehle Schieb- / Rotierbefehle	ADD AND LSL
Programmsteuerbefehle	bedingte Verzweig. unbedingte " Unterprogrammaufruf -ende	BEQ BRA BSR RTS

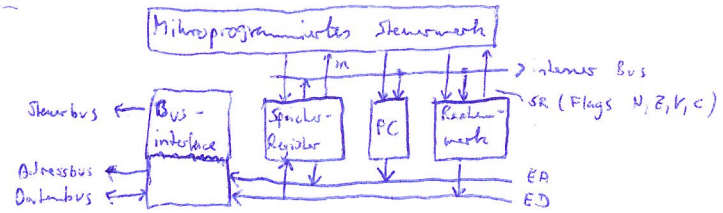
Adressierungsarten: z.B.
- unmittelbar: Operand ist Konstante; in Befehl enthalten
- direkt: Befehl enthält Adresse d. Speicherzelle d. Operanden
- indirekt: Adresse wird indirekt ermittelt (z.B. über Register)

Codierung v. Befehlen: Null-/Ein-/Zwei-/Drei-Adressmaschine: # gibt die Anzahl d. Operanden die ein Befehl adressieren kann an

Bsp. eines mikroprogrammierten Prozessors

Spezifikation eines Minimalprozessors:
- Akkumulatorbasiert (⇒ Einadressmaschine)
- 16 Bit Wortbreite
- 2^{16} Worte; Adressraum ($0000 ... $FFFF) ≙ 128 kByte
- Adressierungsarten: unmittelbar & direkt

Implementierung des Minimalprozessors:



Ausblick: Einschränkungen:
- pro Befehl nur ein ext. Operand verarbeitbar
  - ↳ viele Befehle
  - ↳ hoher Speicherbedarf
  - ↳ lange Programmausführungszeit
- nur ein Wert im Prozessor speicherbar (→ PC)
  - ↳ Operanden müssen im Speicher abgelegt werden, dadurch großer Overhead durch Abspeichern und Laden des Akkumulators
- nur feste, absolute Adresse möglich
  - ↳ kein Zugriff auf Feldelemente
  - ↳ Realisierung von Unterprogrammen nicht möglich

→ Erweiterung: Zweiadressmaschine, Interner Registersatz, indirekte Adressierung } Modellprozessor

Assemblerprogrammierung

Programmiermodell eines Modellprozessors (CISC):

Speichermodell
- Datenbus = 16 Bit = Adressbus
  - ↳ Maschinenworte und Adressen umfassen 16 Bit
- auch auf einzelne Bytes zugreifbar
- max Speicherausbau: 2^{16} Byte = 64 kByte (beschränkt durch # Adressleitungen)
- MSB first
- Ein-/Ausgabegeräte über spez. Speicheradressen (memory mapped I/O)

Registermodell:
- 4 Datenregister
- 4 Adressregister
- PC
- SR